Corrección al teorema de Cook

Juan Manuel Dato Ruiz

La constitución del pensamiento exclusivamente lógico

Resumen.

Según parece, fue Aristóteles el primer hombre en intentar adivinar la estructura y formas que componen un pensamiento riguroso y lógico. Este documento intenta plasmar el resultado de la historia de la computación y de la informática sobre el diseño de un artefacto capaz de llegar a conclusiones lógicas que desafían los límites impuestos por los sistemas neuronales tal como se entiende hoy día en la informática. Asímismo, como un cerebro no tiene porqué ser compatible con un ordenador, nuestro modelo al final deberá evadirse de los sistemas digitales para decantarse con una estructura que nos recuerda más a la programación cuántica.

Porqué nos estamos retrasando

Desde hace ya unos 30 años se ha estado defendiendo una teoría en el mundo de la informática teórica que ha tratado de simplificar enormemente las conclusiones a las que se pueden llegar en este campo. Este es el caso de algunos teoremas que nunca debieron de recibir el nivel de certeza que adquirieron y que, de hecho, por la torpeza de no querer corroborar la calidad de los documentos corre el riesgo de contaminar a los estudios de mecánica cuántica, e incluso a la propia neurología. Estoy hablando del teorema Cook.

No hay que negar el talento inconfundible por parte de una inteligencia y sabiduría como la que tiene Stephen Cook, que tanto ha hecho por la divulgación de los problemas NP, pero en esta primera parte empezaré a dar unos argumentos relativos a la NP-completitud y a la idea de los problemas co-NP (complementary non deterministic polinomial).

La satifiabilidad lógica es el nombre que recibe el problema de saber si una afirmación es o no inconsistente. Una formulación más estricta del problema nos obligaría a tener que plantear bajo qué formato será presentada la fórmula a evaluar, pero si queremos ser más generosos y genéricos bien podríamos decir que de lo que hablamos es de una fórmula bien formada (aquella que conecta variables cuyo valor puede ser verdad o falso con conectores, ya sea and, or ... etc).

Dentro de la extraña terminología usada por los informáticos, una de tantas cosas que se usaron fue el concepto de coNP. Los problemas coNP intenta recordar a todos los problemas que se pudieran presentar complementarios a los llamados problemas NP. Esto es, si éramos capaces de saber en tiempo polinomial (el número de pasos necesarios para dar con la respuesta debe ser inferior a Kx^K para alguna K si x es el tamaño de la entrada, trabajando estas cantidades ya sea en una máquina de Turing o en cualquier tipo de mecanismo que se le parezca – no perderemos mucha rigurosidad si pensamos en un ordenador) si la solución que teníamos preparado servía de contraejemplo a una de las respuestas que plantee el problema (según la definición de Hilbert, un problema sucumbe a ser respondido con un sí o con un no) entonces estábamos ante un NP.

Hasta aquí se podía entender fácilmente: si se puede validar una respuesta con facilidad, entonces es NP. Este es el caso de la satisfiabilidad lógica: si cierta asignación de variables nos lleva a la afirmación de que la fórmula es verdadera, como el cálculo de las sustituciones y aplicación de las tablas de la verdad sobre la expresión nunca excederá la cota polinomial del tamaño de la entrada esto podría servir de contraejemplo ante el problema de la inconsistencia de la fórmula (aquella que dice que se pongan los valores que se pongan la fórmula siempre sería falsa).

Pero ante esta definición aparece una aplicación de lo mencionado antes (coNP). El problema complementario al de la satisfiabilidad nos plantea si la fórmula bien formada es un teorema.

Satisfiabilidad: ¿Existe x: F(x)=True? (de existir x, F no podría ser inconsistente).

Complementario:

¿no Existe x: F(x)=True?

¿todo x: no (F(x) = True)?

¿todo x: no F(x) = True?

¿es no F un teorema?

Es curioso que resultara difícil para algunos establecer una relación entre NP y coNP, sin embargo más adelante entenderemos el porqué. En cualquier caso, hagamos unos pequeños ejercicios de lógica: Si sómos capaces de plantear en una máquina si F(x) = True, entonces también seremos capaces de plantearle a la máquina si ¬F(x) = True, esto es, el conector negador convierte a la fórmula en su negada. Por tanto, nuestra capacidad para dar respuesta a la satisfiabilidad lógica implica que podríamos dar respuesta a ambas preguntas, ocurriendo tres posibilidades:

a) Existe un x que satisface F pero no ¬F. Entonces F es teorema.
b) Existe un x que satisface ¬F, pero no F. Entonces F es inconsistente.
c) Existe un x que satisface F y otro para ¬F. Entonces F no es teorema ni inconsistente.

Por tanto, a nuestra capacidad para saber si una afirmación es teorema lógico se asocia a poder resolver problemas coNP. Y de aquí se deduce trivialmente que todos los problemas lógicos coNP son problemas lógicos NP. Por otro lado, si yo sé que una fórmula es teorema o no, entonces también sé si la fórmula opuesta es o no teorema; por tanto NP = coNP en los problemas de la lógica.

Hubo un libro que hizo un compendio laborioso y muy interesante sobre los problemas NP más importantes, en éste se expuso un curioso resultado: decía que el problema al que llamaba SAT (que no es sino una normalización del problema de la satisfiabilidad expuesto anteriormente) era NP-Co (NP completo) y que su problema problema complementario coNP-co (coNP completo). Esto quería decir que si SAT fuera resoluble en tiempo polinomial (el tiempo en determinar el valor que valide el resultado) entonces todos los problemas NP serían resolubles en dicha cota; y lo mismo pasaba con su complementario y los problemas coNP.

Estas afirmaciones provienen como raíz del teorema de Cook, citado antes, y cuyos corolarios desarrollamos a continuación.

Es posible que alguno considere que la operación de negar una fórmula bien formada es una operación ardua y complicada; sin embargo, en ocasiones, es más bien al contrario. La normalización del problema SAT exige representar la función como producto de sumas de literales (variables que pueden ser afirmadas o negadas), y un desarrollo más pormenorizado podría darnos a entender que si el negado es difícil de calcular, el afirmado lo será fácil (y a la inversa). Por tanto, si barajamos los conceptos nos daremos cuenta del siguiente hecho: las conclusiones a las que hemos llegado sobre la complementariedad de los problemas NP son aplicables también a la completitud, por lo que (si el teorema de Cook es cierto) NP = coNP.

Sin embargo, el objeto de este apartado no es la demostración de que NP sea igual a coNP, ya que, de hecho, no lo es. Por la misma razón por la cual el teorema de Cook nos dice que no es posible encontrar una resolución en tiempo polinomial a una fórmula lógica y asegurar que no todos los problemas NP pueden resolverse en tiempo polinomial, cosa que a mi juicio ya he hecho (y está pendiente de divulgar). El objeto de este apartado es esclarecer al máximo que hay un error en esa endemoniadamente compleja demostración del teorema de Cook, donde explicar el error en la misma exigiría entender la estructura de la máquina de Turing.

Para demostrar mis afirmaciones dispongamos del siguiente problema NP: decir si existe solución a una ecuación diofántica para un vector de valores cuyo módulo sea inferior a un cierto valor K.

Antes de nada explicaré qué significa todo esto: una ecuación diofántica es una igualdad entre dos expresiones de sumas y productos de constantes y variables que toman valores enteros. Asímismo, si cogemos las variables formando un vector con ellas, entonces la segunda condición es que, habiendo determinado un mecanismo para calcular su módulo, que éste sea menor que un cierto K dado de antemano.

Huelga mencionar que una ecuación diofántica es algo que ninguna máquina de turing puede resolver, pues no existe configuración lo suficientemente general capaz de dar solución a cualquier ecuación sin sucumbir a una contradicción en sus pasos; sin embargo el problema es resoluble por una sencilla razón: puede darse uso a una cantidad finita de máquinas de turing que prueben valores, el número de máquinas está acotado por K y su evaluación será polinomial; es por ello que entra dentro de la definición de problema NP.

Dado que este problema es NP, podemos encontrar la expresión de su co-NP:

Problema NP

¿Existe v: expr1(v) = expr2(v) y |v|<=K?

Problema co-NP

¿¬Existe v: expr1(v) = expr2(v) y |v|<=K?

¿todo v: ¬(expr1(v) = expr2(v) y |v|<=K)?

¿todo v: expr1(v) <> expr2(v) o |v|>K)?

¿todo v: expr1(v) = expr2(v) ->|v|>K)?

¿para cada posible v que cumpla una ecuación diofántica su módulo excede de K?

Esto exige que infinitas máquinas de Turing puedan resolver una ecuación diofántica cada una con el fin de dar con la respuesta.

Por tanto, si hay un coNP que no es planteable como problema, tampoco podríamos afirmar NP = coNP (pues no es posible validar lo que no se puede resolver). De lo que deducimos que el teorema de Cook no puede ser coherente.

El error de Cook

En el libro “Computers and Intractability” de Michael R. Garey & David S. Johnson tenemos una transcripción del famoso teorema de Cook, descrito en 1971. Cuando vemos los resultados aquí expuestos nos vemos ante la tesitura de encontrar una contradicción: por un lado se afirma que los problemas contenidos en la clase NP no están todos contenidos en la clase P y, por otro lado, se afirma que un problema que es NP completo (SAT) se encuentra en P. No debemos descartar en ningún momento que aquí hay que subsanar un error existente pero, ¿dónde está el error? Viendo de forma tangible el lugar que ocupan las afirmaciones expuestas en esta documentación, mi intención apunta a la necesidad de abrir un nuevo paradigma en el mundo de las matemáticas y la informática, debido a un error muy humano: aceptar una demostración que no es válida.

Para entender por qué puedo afirmar de que el teorema de Cook es falso podemos dirigirnos a su propia transcripción: El teorema de Cook, en el libro citado, comienza definiendo la clase NP-Completo (page 37) como el conjunto de los problemas NP en los absolutamente el resto de los problemas NP pueden verse transformados mediante un código que no exceda en tiempo polinomialmente (con respecto al tamaño de la entrada). De esta manera, si un NP-completo pudiera resolverse polinomialmente, entonces todos los problemas NP tendrían una implementación (configuración de Máquina de Turing) polinomial; para así poder decir que NP=P.

Dicho de esta manera la definición suena muy jugosa y, aún más, hasta pretenciosa; pero claro, era objeto de Cook poder demostrar que existía al menos un problema planteable dentro de los NP que cumpliera ese requisito. La demostración de la existencia de ese problema NP-completo inicial se da en el propio Teorema de Cook (page 39).

El razonamiento que hace uso el Sr. Cook en este teorema consiste en afirmar que el problema de la satisfiabilidad lógica puede ser codificado mediante algún esquema razonable a través de un lenguaje L_{SAT}, entendiendo que este lenguaje nos discrimina las entradas que consiguen una afirmación booleana verdadera dentro de la fórmula que representa. Esto es, a través de una Máquina de Turing No Determinística podría resolverse en una cota de tiempo polinomial si la fórmula descrita en el lenguaje satisface o no la fórmula.

Una vez presentado L_{SAT}, ahora se trata de poder asegurar que cualquier lenguaje L contenido en NP pudiera ser resuelto una vez soluble L_{SAT} dentro de la cota fijada. Para lo cual hablaríamos de una función $f_L(x)$ cuya propiedad es que reconocerá cualquier x en L sii f_L contiene una asignación que satisfaga a la fórmula lógica. Entendemos, por tanto, que x es una entrada aceptada por cualquier NP y que posee un vínculo a través de f hacia SAT, donde resolviendo dicho vínculo podríamos resolver en esa cota de tiempo cualquier NP.

El trabajo de esta función consistiría en mapear cada una de las posibilidades en su correspondiente Máquina de Turing y, considerando la cota polinomial, restaría poder demostrar la existencia de una fórmula bien formada que equivalga a la potencia de dicho problema. Y, de hecho, es así como continía el proceso de demostración: Debido a que la

máquina es no detérminística la cota estará fijado por un polinomio que toma valor a partir del tamaño de la entrada (page 40) , por tanto, asevera S. Cook “This will enable us to describe such a computation completely using only a limited number of Boolean variables and a truth asigment to them”. En este punto ya uno debe decir: “sí, en teoría” ¿Podemos asegurar que para cualquier problema NP que nos queramos plantear (por muy duro que sea) siempre seremos capaces de encontrar una fbf que represente el mismo problema?

Cuando Cook continúa con la afirmación nos enumera una “cantidad polinomial” de proposiciones completamente indefinida; desde un punto de vista formalista un matemático nos diría que, efectivamente, las proposiciones quedan definidas, sin embargo debemos ser más rigurosos: ¿existe una correspondencia entre el problema original y la fórmula bien formada que se pretende contruir? Es decir, ¿es constructible? Más en concreto: a medida que vayamos construyendo la fórmula con todas sus proposiciones sólo podremos reconocer la existencia de tales proposiciones como predicados de orden 0 en la medida en la que vayamos ejecutando el problema L dentro de la máquina que lo resuelva. Esto es, para poder construir la fórmula bien formada antes debemos conocer la solución del problema.

Ese es, básicamente, el error del teorema de Cook: para poder constituir el primer NP-completo antes debe resolver cada unos de los posibles problemas NP. Obviamente es insostenible, porque una vez resuelta la satisfiabilidad lógica no tendremos garantías de encontrar un código que nos convierta el problema en cualquier otro NP.

Por otro lado, aunque el teorema de Cook aseguraba que existía dicho código, desconozco ninguna equivalencia capaz de aprovechar los pequeños avances que se hayan hecho sobre la lógica para aprovecharlos en el mundo de la matemática en general; hay que imaginarse hasta qué punto algo así habría sido beneficioso.

De hecho, es cuestión de ir un poco más allá: ¿acaso no seríamos capaces de plantearnos la resolución de un problema del Principia Matemática como un problema acotable en el tiempo? El teorema de Cook no pone límites a la cota, sólo hace referencia a que la cota es polinomial; bien, cambiemos la palabra polinomial por exponencial, o incluso acotada sin más (que no se cuelga). Si dispusiéramos de un enunciado descrito desde los axiomas del Principia Matemática, el enunciado sería demostrable en un tiempo acotado por alguna máquina de Turing, consiguiendo así una omega coherencia, siempre y cuando aplicáramos algún algoritmo de unificación razonable (como el de Robinson), ya que nuestra intención es llegar a un objetivo. Es entonces que si combinamos los razonamientos del teorema de Cook con su aplicabilidad sobre el Principia Matemática, siempre encontraríamos una fórmula a satisfacer dentro de la lógica de orden 0 que satisfaría una formulación del principia matemática; es decir, esto contradice los dos famosos resultados de Gödel: el de la completitud y el de la incompletitud. Es por ello que el teorema de Cook no puede ser aceptado desde un punto de vista teórico.

El problema de la igualdad

Para poder demostrar que las clases son diferentes antes debemos abordar los distintos tipos de problemas que nos podemos encontrar. Esto es, cuando generamos una correspondencia entre dos datos cualquier conocimiento que tengamos sobre uno repercutirá sobre el conocimiento que tendremos del otro. Este concepto es muy importante pues la existencia de una correspondencia nos obliga a restringir la idea que tenemos de lo que es una asignación: si declaramos un vínculo entre dos datos entonces nos vemos obligados a rechazar algunas combinaciones que se escapen de su invariante como sistema.

Es por ello que podemos definir sistemas donde el conocimiento de unos datos nos podrían llevar a desconocer otros datos. Sin embargo antes debemos constituir ese tipo de máquinas de una manera constructiva, es decir, es objeto de este documento demostrar su existencia creando un sistema que funcione de esa manera.

Así que antes de empezar con las aplicaciones prácticas (firmas digitales o creación de canales seguros a partir de canales inseguros) debemos crearnos una máquina de juguete (debido a que el álgebra necesaria para crear esos ingenios pueden complicar la demostración de la diferencia entre P y NP).

Así decimos que existe una correspondencia entre tres datos que conoceremos: X, Y y Z. Pero para demostrar la existencia de esa correspondencia antes vamos a construir los parámetros que la conforman: A, B, C y D.

Para ello nos imaginaremos que el mundo es como un espacio de Hilbert (todo finito y acotado) en nuestro universo todo es plano y está cuantificado. Es por ello que la única idea de correspondencia que debe existir se dará a través de cuadrados latinos irremediablemente. De esta manera podemos asociar X con Y diciendo que X es la fila del cuadrado latino e Y el valor posicionado dentro de la fila y alguna columna del cuadrado latino. Es por ello que para que a X se le asocie un Y específico necesitaremos un parámetro, al que llamaremos D, que ocupará la columna.

Ahora se trata de volver a operar: Cogemos el parámetro D y lo combinamos con la Y (fila y columna) dentro de otro cuadrado latino, de manera que nos devuelva X. El valor que encierra el misterio de esta posibilidad lo ubicaremos en el parámetro C. Podemos decir, de esta manera, que el número de cuadrados latinos está encerrado en el parámetro C ya que el cuadro que nos permite relacionar una fila con el valor para dar con la columna no sólo puede ser un cuadrado latino sino, de hecho, varios posibles. La deducción de esto es trivial.

Teniendo dos valores (C y D) podemos darle valor a A y a B. Para ello interpretaremos el valor de X y de C como si representaran permutaciones de valores pares. Acto seguido calculamos:

$$A = X_0 – C_0$$

Después cogemos los valores Y, D y X de nuevo, y los interpretamos como si fueran permutaciones de valores impares para así calcular:

$$B = Y_1 – D_1 – X_1$$

Llegados a este punto entenderemos que conociendo X, X_0 o X_1 podríamos saber el valor de los otros dos. Sin embargo, no nos quedaremos tan a gusto con nuestros cálculos si no somos capaces de incluirle una dimensión más a nuestro sistema. Para ello definimos el valor de Z como el resultado de aplicar al cuadrado latino original la A con la D.

Y con esta sopa de letras ya estamos dispuestos para emprender nuestro viaje al estudio de las correspondencias que se "ocultan".

Esto es, conocidos los valores X, Y, Z ¿qué parámetros podríamos llegar a conocer en un tiempo acotable? En cuanto a que el universo es de Hilbert la cota es de vital importancia, pues si no estuvieran acotados es posible que nuestros planteamientos deberían cambiar. Sin embargo huelga citar que con sólo conocer X, Y y Z no seremos capaces de deducir absolutamente ningún parámetro interior. Sin embargo, siempre podríamos encontrar soluciones partiendo del conocimiento de A, para saber C y sólo C. Así como si partimos del método de conocer B entonces sólo podremos conocer el valor de D y sólo D. Esto es debido a que entre D y C existe una multidependencia, cosa que se rompe en el momento en el que sabemos del valor de Z como resultado de conectar A con D. Es por ello que para que sea coherente la información inicial (X,Y,Z) con el método escogido, (método B) nos vemos ante la tesitura de que no podemos disponer de conocimiento alguno sobre el valor de C aun existiendo una correspondencia.

Una vez conocidos los valores acotados A,B,C,D siempre podrían validarse las coordenadas conocidas (X,Y,Z) conllevando a que el proceso de validación sí es fácilmente acotable y, por tanto, puede fijarse este problema como de clase NP.

Es decir, conocida una trinca X,Y,Z que se relacione a través de un cuadro latino de referencia y dos parámetros desconocidos C y D, si nos planteamos si el valor fijado en B nos genera una posible A que cuadre todas las relaciones faltantes nos daremos cuenta de que el procedimiento para corroborarlo exige la comprobación de cada una de las posibles formaciones de un cuadrado latino, sabiendo que el número excede siempre al cardinal de todos los valores que puede adquirir la entrada.

Es por ello que no es admisible la acotación polinomial en una formación como ésta.